DOCUMENT RESUME

ED 392 425 IR 017 716

AUTHOR DiBiase, Julie; Eisenberg, Michael
TITLE Mental Imagery in the Teaching of Functions.
SPONS AGENCY National Science Foundation, Arlington, VA.
PUB DATE 95
CONTRACT IRI-9258684; RED-9253425
NOTE 8p.; In: "Emerging Technologies, Lifelong Learning, NECC '95"; see IR 017 705.
PUB TYPE Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Case Studies; *Comprehension; Curriculum Development; *Curriculum Guides; Elementary Secondary Education; *Functions (Mathematics); Instructional Innovation; Mathematical Concepts; *Mathematics Curriculum; Mathematics Instruction; Visualization
IDENTIFIERS *Mental Imagery; *Multimodal Methods

ABSTRACT

Few would argue that students struggle with understanding and representing functions. There are many theories on why the abstraction of mathematical processes poses such cognitive difficulty, but so far functional programming is most often taught in a manner that discourages the use of visual intuition. The paper, by contrast, describes a curriculum for students ages 9 to 15 that addresses the concept of higher-order functions by means of multi-modal imagery building. It discusses the mathematical importance of higher-order functions, outlines the motivation behind the curriculum, and offers a case study of a fifth-grade girl learning functional arguments via computer. (Contains 24 references.) (Author/BEW)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)
☐ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED 392 425

Mental Imagery in the Teaching of Functions

Paper presented at the NECC '95, the Annual National Educational Computing Conference (16th, Baltimore, MD, June 17-19, 1995.

IR017716

BEST COPY AVAILABLE

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Donella Ingham
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

paper

Mental Imagery in the Teaching of Functions

Julie DiBiase
Department of Computer Science and Institute of Cognitive Science
University of Colorado, Boulder
Boulder, CO 80309-0430
(303) 492-1218
jules@anchor.cs.colorado.edu

Michael Eisenberg
Department of Computer Science and Institute of Cognitive Science
University of Colorado, Boulder
Boulder, CO 80309-0430
(303) 492-8091
duck@sigi.cs.colorado.edu

Key words: functions, higher-order procedures, functional programming, mathematical imagery, mathematical manipulatives

Abstract

Functional programming is traditionally taught in a manner that discourages the use of visual intuition. This paper, in contrast, describes a curriculum that addresses the concept of higher-order functions by means of *multi-modal imagery building* We discuss the mathematical importance of higher-order functions; outline the motivation behind our curriculum; and provide a case study of a fifth-grade student learning about functional arguments.

Introduction

> "In mathematics... we find two tendencies present. On the one hand, the tendency toward *abstraction* seeks to crystallize the *logical* relations inherent in the maze of material that is being studied.... On the other hand, the tendency toward *intuitive understanding* fosters a more immediate grasp of the objects one studies, a live *rapport* with them, so to speak, which stresses the concrete meaning of their relations...." [Hilbert 1932]

Historically, functional programming and higher mathematics have been characterized by an almost relentlessly textual and axiomatic style of teaching; the educational tradition for these topics is largely devoid of the rich "manipulatives" and visual images that characterize much of the most creative work in basic math education [Davidson 1977; Montessori 1956]. We would like to narrow the gap between these two educational traditions by applying the results of mental imagery research to the teaching of abstract mathematics and computer science concepts. Accordingly, we have undertaken an in-depth study of an extremely important "higher-order" concept common to both higher mathematics and computer science—namely, the idea of functional data objects.

Our curriculum is designed to facilitate learning through "multi-modal imagery building." By exposure to multiple visual and concrete representations of an abstract concept, the student is encouraged to build up an increasingly complete and accurate mental image of that "target" concept. For any particular target concept, building the imaging curriculum involves (1) researching student problem areas with the concept; (2) creating a taxonomy of those problem areas; and (3) building tools that suggest visual images aligned with the cognitive difficulties of each problem area identified.

Before providing a detailed case study of how we designed and executed a curriculum for the concept of functions as data objects, it is worth motivating our interest in this notion from both historical and pedagogical perspectives.

Numbers, Functions, and Mental Imagery

> "We shall see... that the "abstraction" of the number sequence from the things counted created great difficulties for the human mind. We need only ask ourselves: how would we count if we did not possess this sequence of remarkable words, 'one,' 'two,' three,' and so on? ... [O]ne achievement of our number sequence is its independence of the things themselves. It can be used to count *anything*." [Menninger 1969]

Few would argue that students have difficulty in understanding and representing functions. Indeed, similar challenges may be identified historically in the genesis of the number system: early civilization's concept of number reappears developmentally in present-day children's initial concept of number. The central challenge, both for modern children and ancient adults, lies in separating the objective and abstract nature of number from the "thing to be counted" [Menninger 1969].

While early civilizations outgrew (and children likewise outgrow) their misconceptions of number objects, the same is not true for the "objectification" or abstraction of processes. One pertinent study indicates, intriguingly, that children exhibit a linguistic reluctance towards the symbolic recognition of "active entities": in a series of interviews with Argentinean children aged 4-6 who had not previously experienced written language, Ferreiro [1978] found that the subjects intuitively believed that nouns could be described in written words (e.g.. the word "Daddy" in "Daddy kicks the ball."), but the same was not true for verbs (e.g. the word "kicks" in the same sentence).

Ferreiro's results suggest, by analogy, that students' difficulties in representing a functional object may have deep roots in the characteristic verb/noun distinctions of natural language. And indeed, contrary to the adult proficiency that eventually develops over the concept of number, the object concept of function remains even today the province of the mathematical elite. The difficulty that students experience with this notion is exacerbated by opaque mathematical notation (e.g., the "prime" notation for the differentiation operator) and is confirmed by other researchers. Harel and Dubinsky [1992] propose a cognitive model in which the "object" concept of function is the most advanced stage in the understanding of functions; Cuoco [1995] has also treated this same issue as central to the understanding of higher mathematics.

It would be reasonable, then—based on these observations—to address the cognitive difficulties surrounding the understanding of functional data objects with the same types of "imagery-enhancing" manipulatives that characterize the teaching of the number concept. Nonetheless, while there is a large and growing body of research linking mental imagery and mathematical cognition [Seron, Pesenti *et al.* 1992; Galton 1883; Hadamard 1944; Kaufmann 1990; Antonietti 1991], comparatively little of this work has concentrated on the abstract domain of functional objects. This gap is particularly troubling since abstract domains of this type are precisely those in which imagery is most urgently required for understanding, as argued by Kaufmann:

> "Conditions where imagery is most readily available may not be the same as the conditions where it is most strongly needed. Imagery may be most readily available in concrete tasks, but conceivably is most highly needed in abstract task-environments."[Kaufmann 1988]

Abstracting Functions as Data Objects

> "In studying the action of the Analytical Engine, we find that the peculiar and independent nature of the consider-

ations which in all mathematical analysis belong to *operations*, as distinguished from the *objects operated upon* and from the *results* of the operations performed upon those objects, is very strikingly defined and separated."[Lovelace 1842]

The "naive ontology" reflected by the above quotation from Countess Lovelace reflects a sharp intuitive distinction between "actions" and "objects", much like the distinction exhibited by the subjects in Ferreiro's aforementioned study. Indeed, the Countess's ontology parallels—with fascinating accuracy—that of the subjects we have studied [Eisenberg *et al.* 1987; DiBiase 1994]. For these subjects, data and functions are disjoint types of things. Ideally we would like students to acquire a cognitive model in which functions and numbers are two subsets of a single class of "first class data objects", whose properties are as follows (cf. Stoy [1977]):

1.) First class objects can be named.
2.) First class objects can be passed as arguments to functions.
3.) First class objects can be returned as results of function calls.
4.) First class objects can be combined to form complex data structures.

In a study of MIT undergraduates taking an introductory Scheme course,Eisenberg *et al.* [1987] documented the difficulties that students encounter in viewing functions as data objects along these lines. For instance, more than half of the MIT students interviewed reasoned incorrectly (at least at first) about the evaluation of the following Scheme expressions:

```
(define (create-subtracter n) (lambda (x) (- x n))

(define (apply-to-5 f) (f 5))

(apply-to-5 create-subtracter)
```

These results have since been confirmed by studies conducted with University of Colorado graduates and undergraduates: over half of the 40 students responding to questionnaires showed some form of misapplication of the functional data object concept. [DiBiase 1994].More important than the fact that errors occur is the fact that they are remarkably consistent in structure; we will return to this point shortly.

Applying An Imagery-Based Curriculum to Functional Data Objects: a Case Study

As noted earlier, our curriculum is designed to encourage the development of appropriate visual imagery for the concept of functional data objects. The curriculum is a 10-20 hour program of individual instruction offered to students who have ranged in age from 9 to 15; students work in weekly 1-2 hour sessions. Structurally, the curriculum focuses on three pedagogical methods:

Re-intuition

Students explore and challenge their preconceptions about the roles that objects play using "situational world analogies" (cf. Vosniadou and Shcommer [1988]) conducted through discussion and written questionnaires.

Concretization

Students use a mixture of representations—computational (programming), written (visual representations on paper and whiteboard) and concrete (hand made objects that students can touch and manipulate) to exercise skills of "objectifying" abstract concepts (cf. [Papert 1980; Montessori 1956; March 1977]).

Contextualization

Students learn to program in SchemePaint, a graphics application employing an embedded (Scheme) functional programming environment; the environment is designed to make extensive use of rich graphical programming constructs and thus to serve as a potentially interesting venue in which to learn programming (cf. [Paivio 1988; Owen 1986]).

In the remainder of this section we provide a case study of an elementary school student who undertook this curriculum.

A Case Study

BN, a 10 year old female in fifth grade, had no prior programming experience: her previous computer experience was limited to word processing and game playing. The following discussion represents her first 5 hours of learning to program in SchemePaint. It focuses on the acquisition of the notion that functions can be arguments to other functions (Stoy's property 2)–an idea which necessitates a strong object concept of function.

We began the project by spending the first hour discussing her concept of computers and active and passive entities in the non-computational world (exercising the re-intuition methodology). She viewed the computer as both an active and passive entity (Cf. [Turkle 1984]). By discussing the characteristics of people, computers, books, tables and other objects in the world, we went on to convince ourselves that objects can be named and can have properties of an active and/or passive nature. We also discussed the mathematical concept of a variable, with which she was previously unfamiliar. The purpose of focusing on these concepts in a non-computational domain was to provide a mental framework for the computational analog (Cf.[Vosniadou and Shcommer 1988; Gentner and Stevens 1983]).

BN retained this conception of objects in the world in the next session, when we extended the notion to computational objects. She was introduced to number-objects and procedure-objects in SchemePaint, and was given an explanation of how the "Scheme-engine" (i.e. interpreter) maintains and utilizes a huge table of information about these objects. This first introduction to data objects is of a written textual nature. In Figure 1, we show two example objects in the "name/object" space as they are first seen by the student:

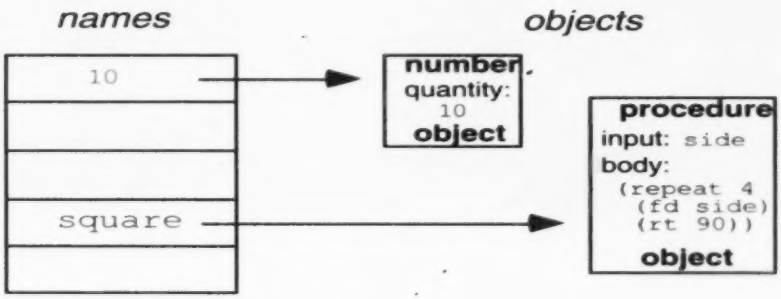

Figure 1: The name/object table

In order to elaborate the idea of how the Scheme-engine operates in the name/object space, we used concrete procedure and number objects (represented by small hand-sewn felt pillows). In virtue of their manipulability, these objects help to reify the notion that everything in the Scheme world, including procedures, have the same object-status. Further, the fact that the physical objects shared some properties (all objects are rectangular in shape and "puffy" in feel) and some different properties (color and size) helps to reify the notion that computational objects in Scheme have both shared properties (e.g. those described by Stoy, noted above) and different properties (numbers can be added, procedures can be executed, etc.).

The value of the manipulatives was clear when, during the following session, BN was asked to explain the invocation of a procedure call. Although the concrete objects were not presented for use during this latter session, BN motioned throughout her explanation as if she were holding objects in her hands. She consistently spoke about number-objects and procedure-objects as "floating around".

In the second session, BN wrote her first procedure to draw a square (thereby framing or "contextualizing" the experience within the domain of graphic arts):

```
(define (square side)
  (repeat 4
          (fd side)
          (rt 90)))
```

During the third meeting, she wrote her second procedure to make a rectangle and then, with a little mathematical assistance, a third procedure to make a hexagon:

```
(define (hexagon side)
  (repeat 6
          (fd side)
      (rt 60)))
```

Figure 2a shows the result as viewed in the graphics window.

Figure 2:
a. The product of BN's original hexagon procedure.
b. The product of BN's revised hexagon procedure which takes a function as its argument.

During the fourth session, it was proposed to her that we might like to modify the sides of the hexagon to do other things other than just go forward. BN was shown examples of six-sided figures that had varying shapes on each side (e.g. Figure 2b. shows a hexagon-like shape in which the sides were made using a "zig" procedure rather than the "fd" procedure), and was then asked to write a procedure that could achieve this or any number of other similar effects, keeping the side length con-

stant. She chose to write it at the dry-erase board. She worked out the problem almost entirely on her own. After five minutes of deliberating and modifying, she had correctly completed the problem. Below is a transcript of her modification process, with student-teacher interaction included:

STEP 1:

```
(define (hexagon 20)        [line 1]
          (repeat 6         [line 2]
                  (fd 20)   [line 3]
                  (rt 60)))  [line 4]
```

[BN: "This isn't right. I'm just thinking"]

STEP 2: [working with line 3 for STEPS 2-4]

```
(? 20)
```

STEP 3:

```
(fd pro 20)
```

STEP 4:

```
(pro 20)
```

[At this point BN was told that she was doing well. She seemed stuck for a few minutes, and—after responding to the suggestion that she could ask for a hint—she was asked to think about how the computer would know what "pro" was.]

STEP 5: [working with line 1]

```
(define (hexagon pro 20)
```

[BN was informed that she was close, and was asked how she would use the procedure. In trying to come up with a correct invocation, she realized her mistake and completed the correct version.]

FINAL:

```
(define (hexagon pro)
          (repeat 6
                  (pro 20)
                  (rt 60)))
```

She was then able to explain and use the procedure correctly. Thus, after 3-4 hours of instruction on SchemePaint, BN was able to successfully and independently make use of a function as a data object without an excessive amount of mental effort.

In the last of the five sessions, BN was asked to explain the implementation details of all the procedures she had written. She was given the option to use the manipulatives to explain the invocations. She did not opt to do so, but rather motioned again as if she were holding objects. BN's explanation was correct to an adequate degree of detail, given her brief exposure to the concepts. When asked to explain using the concrete object, the explanation was again correct.

It can be concluded that BN's high-level model of procedures supports an object concept. It became clear from observing her explanations that, even in the absence of the physical objects, BN still visualized the objects very concretely as she pantomimed the physical act of gripping something. It was interesting to note that when BN was asked to write a new procedure she did not just write the Scheme code (on the board): she began by drawing a box on the board and labeling it a procedure-object (one of the other written visual representations we had discussed), then inserting the code in the box. Thus, BN had learned to work comfortably with at least two different ways of explaining the same concept, both involving some sort of object-imagery of the subject (either pictorial or physical).

Summary of Results; Future Research Directions

To date, our curriculum has been employed with ten students. Space limitations preclude the inclusion of additional case studies similar to the one above, but our observations may be summarized as follows:

a. Properties 1-3 (cf. [Stoy 1977]), represent strictly increasing levels of difficulty for students.
b. Students have trouble with anonymous objects of many types (i.e. the problem is not limited to anonymous procedural objects).
c. In trying to progress to a mental model that includes property 3, students almost uniformly mistake correct code as dysfunctional because of reports that variables are "missing" from lambda expressions.
d. Students have trouble with the lambda notation because its semantic content is obscure and suggests few fruitful analogies.

Notably, these observations apply with a high degree of consistency to all of the students that we have worked with; and they are also consistent with the informal reports provided by teachers of functional programming.

Our experience to date with an "imagery-building curriculum" for procedural objects is encouraging, and suggests a variety of fascinating questions for continuing and future research. Certainly, we would like to know whether students' correct use of procedural objects in a functional language such as Scheme can transfer to a deeper (or easier) understanding of such related abstract mathematical concepts as "function spaces". More generally we would like to extend the techniques of imagery-building to a variety of similarly abstract topics in computer science and higher mathematics—e.g., continuations, recursion, and group theory. We would hope to improve pedagogy in these topics—as in the case of higher-order functions—by starting with an identification of the problems that students have in representing abstract concepts, and proceeding to explore how those particular problems may be addressed through a rich repertoire of methods for enhancing mental imagery.

Acknowledgments

Many people have contributed to this work; we especially wish to thank Al Cuoco, Wally Feurzeig, Gerhard Fischer, Mitchel Resnick, and Franklyn Turbak for their insights and conversation. This work has been supported in part by NSF grants RED-9253425 and a Young Investigator award IRI-9258684. Our thanks also to Apple Computer Co. for their generous donations of equipment.

References

Antonietti, A. (1991). Why Does Mental Imagery Facilitate Problem Solving? In R. Logie and M. Denis, eds. *Mental Images in Human Cognition* (211-227). Amsterdam: North-Holland.

Cuoco, A. (1995). Computational Media to Support the Learning and Use of Functions. To appear in C. Hoyles, ed. *The Design of Computational Media to Support Exploratory Learning*. Heidelberg: Springer-Verlag.

Davidson, P. (1977). Rods Can Help Children Learn at all Grade Levels. *Learning*, 6(3), 87-89.

DiBiase, J. (1994). Higher-Order Procedures: Building Curriculum to Shape Cognitive Models. (Unpublished manuscript.) University of Colorado, Boulder.

Eisenberg, M., M. Resnick, and F. Turbak. (1987). Understanding Procedures as Objects. In G. Olson, S. Sheppard, and E. Soloway, eds. *Empirical Studies of Programmers: Second Workshop* (14-32). Norwood, NJ: Ablex.

Ferreiro, E. (1978). "What is in a Written Sentence? A Developmental Answer." *Journal of Education, 160*, 25-39.

Galton, F. (1883). *Inquiries into Human Faculty and Its Development*. London: Macmillan.

Gentner, D. and A. Stevens (1983). *Mental Models*. Hillsdale, NJ: Lawrence Erlbaum.

Hadamard, J. (1944). *The Psychology of Invention in the Mathematical Field*. New York: Dover.

Harel, G. and E. Dubinsky, Ed. (1992). *The Concept of Function: Aspects of Epistemology and Pedagogy*. Mathematical Association of America Notes and Reports Series.

Hilbert, D. and S. Cohn-Vossen (1932). *Geometry and the Imagination*. New York: Chelsea.

Kaufmann, G. (1988). Mental Imagery and Problem Solving. In M. Denis, J. Engelkamp, and J. Richardson, eds. *Cognitive and Neuropsychological Approaches to Mental Imagery* (23-55). Dordrecht: Martinus Nijhoff.

Kaufmann, G. (1990). Imagery Effects on Problem Solving. In P. Hampson, D. Marks, and J. Richardson, eds. *Imagery: Current Developments* (169-196). London: Routledge.

Lovelace, A. (1842). Notes on the Analytical Engine. *Charles Babbage and His Calculating Engines: Selected Writings by Charles Babbage and Others* (225-295). New York: Dover.

March, R. (1977). Georges Cuisenaire and his Rainbow Rods. *Learning*, 6(3), 85-86.

Menninger, K. (1969). *Number Words and Number Symbols: A Cultural History of Numbers*. New York: Dover.

Montessori, M. (1956). *The Child in The Family*. New York: Avon.

Owen, D. (1986). Naive Theories of Computation. In D. Norman and S. Draper, eds. *User Centered System Design: New Perspectives on Human-Computer Interaction* (187-200). Hillsdale, NJ: Lawrence Erlbaum.

Paivio, A. (1988). Basic Puzzles in Imagery Research. In M. Denis, J. Engelkamp, and J. Richardson, eds. *Cognitive and Neuropsychological Approaches to Mental Imagery* (3-16). Dordrecht: Martinus Nijhoff.

Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books.

Seron, X., M. Pesenti, et al. (1992). "Images of Numbers, or 'When 98 is upper left and 6 sky blue'." *Cognition, 44*, 159-196.

Stoy, J. S. (1977). *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. Cambridge: MIT Press.

Turkle, S. (1984). *The Second Self: Computers and the Human Spirit*. New York: Simon and Schuster.

Vosniadou, S. and M. Shcommer (1988). "Explanatory Analogies Can Help Children Acquire Information from Expository Text." *Journal of Educational Psychology 80*(4), 524-536.